

This Page Is Inserted by IFW Operations  
and is not a part of the Official Record

## **BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning documents *will not* correct images,  
please do not report the images to the  
Image Problem Mailbox.**

**What is Claimed is:**

- 1        1. A method of identifying an errant process in a computer system, the method
- 2           comprising:
  - 3              detecting an error;
  - 4              storing a physical address of an errant process that caused the error;
  - 5              storing an execution instruction pointer (IP) in an interruption instruction pointer (IIP);
  - 6              determining a first virtual address from an operating system mapping table;
  - 7              determining a second virtual address from a translation look-aside buffer; and
  - 8              identifying the errant process, if the physical address and the second virtual address
  - 9              match the physical address and the first virtual address.

- 1        2. The method of claim 1, further comprising:
  - 2              determining whether the physical address of the physical memory location is known;
  - 3              determining in which code section the errant process is located, if the physical memory
  - 4              location is known;
  - 5              resetting the processor, if the physical memory location is in one of a critical section and
  - 6              an unknown section of the code; and

7            terminating the errant process based on a level of sharing of the physical memory location  
8        and whether the IIP associated with the errant process is precise, if the physical memory location  
9        is in a non-critical section of the code.

1            3.        The method of claim 2, said terminating the errant process based on a level of  
2        sharing of the physical memory location and whether the IIP associated with the errant process is  
3        precise, comprising:

4            determining whether the physical memory location is one of a global, a shared and a  
5        private physical memory location;

6            determining whether the IIP associated with the errant process is precise;

7            terminating the errant process, if the physical memory location is global and the IIP is  
8        precise; and

9            resetting the processor, if the physical memory location is global and the IIP is imprecise.

1            4.        The method of claim 2, said terminating the errant process based on a level of  
2        sharing of the physical memory location and whether the IIP associated with the errant process is  
3        precise, comprises:

4            determining whether the physical memory location is one of a global, a shared and a  
5        private physical memory location;

6           determining whether the IIP associated with the errant process is precise;

7           terminating the errant process, if the physical memory location is shared and the IIP is

8       precise; and

9           determining whether the physical address can be used to indicate the errant process, if the

10      physical memory location is shared and the IIP is imprecise; and

11          terminating the errant process, if the physical address can be used to indicate the errant

12      process; otherwise, resetting the processor.

1           5.       The method of claim 2, said terminating the errant process based on a level of

2      sharing of the physical memory location and whether the IIP associated with the errant process is

3      precise, comprises:

4           determining whether the physical memory location is one of a global, a shared and a

5      private physical memory location;

6           determining whether the IIP associated with the errant process is precise;

7           terminating the errant process specified by the precise IIP, if the physical memory

8      location is private and the IIP is precise; and

9           determining whether the physical address can be used to indicate the errant process, if the

10     physical memory location is private and the IIP is imprecise;

11            terminating the errant process, if the physical address can be used to indicate the errant  
12            process; otherwise, resetting the processor.

1            6.        The method of claim 1, said storing the physical address of the errant process that  
2            caused the error comprising:  
3                    storing the physical address of the errant process in a memory register.

1            7.        The method of claim 1, said determining the first virtual address from the  
2            operating system mapping table comprising:  
3                    finding a physical address entry in the operating system mapping table that matches the  
4            physical address; and  
5                    reading the first virtual address from the matching physical address entry, if the matching  
6            physical address entry is found.

1            8.        The method of claim 1, said determining the second virtual address from the  
2            translation look-aside buffer comprising:  
3                    finding a physical address entry in the translation look-aside buffer that matches the  
4            physical address; and

5 reading the second virtual address from the matching physical address entry, if the  
6 matching physical address entry is found.

1 9. The method of claim 1 said execution instruction pointer comprises:

2 an instruction pointer that existed at the time the error is detected.

1 10. The method of claim 1, said operating system mapping table comprises one of:

2 a buffer; and

3 a cache array.

1 11. A machine-readable medium having stored thereon a plurality of executable

2 instructions, the plurality of instructions comprising instructions to:

3 detect an error;

4 store a physical address of an errant process that caused the error;

5 store an execution instruction pointer (IP) in an interruption instruction pointer (IIP);

6 determine a first virtual address from an operating system mapping table;

7 determine a second virtual address from a translation look-aside buffer;

8        identify the errant process, if the physical address and the second virtual address match  
9        the physical address and the first virtual address.

1            12.      The machine-readable medium of claim 11, further comprising instructions to:

2        determine whether the physical address of the physical memory location is known;

3        determine in which code section the errant process is located, if the physical memory  
4        location is known;

5        reset the processor, if the physical memory location is in one of a critical section and an  
6        unknown section of the code; and

7            terminate the errant process based on a level of sharing of the physical memory location  
8        and whether the IIP associated with the errant process is precise, if the physical memory location  
9        is in a non-critical section of the code.

1            13.      The machine-readable medium of claim 12, said terminate the errant process  
2        based on a level of sharing of the physical memory location and whether the IIP associated with  
3        the errant process is precise instruction comprising instructions to:

4        determine whether the physical memory location is one of a global, a shared and a private  
5        physical memory location;

6        determine whether the IIP associated with the errant process is precise;

7 terminate the errant process, if the physical memory location is global and the IIP is  
8 precise; and

9 reset the processor, if the physical memory location is global and the IIP is imprecise.

1       14. The machine-readable medium of claim 12, said terminate the errant process  
2 based on a level of sharing of the physical memory location and whether the IIP associated with  
3 the errant process is precise instruction comprising instructions to:

4       determine whether the physical memory location is one of a global, a shared and a private  
5 physical memory location;

6       determine whether the IIP associated with the errant process is precise;

7       terminate the errant process, if the physical memory location is shared and the IIP is  
8 precise; and

9       determine whether the physical address can be used to indicate the errant process, if the  
10 physical memory location is shared and the IIP is imprecise; and

11       terminate the errant process, if the physical address can be used to indicate the errant  
12 process; otherwise, reset the processor.

- 1        15. The machine-readable medium of claim 12, said terminate the errant process  
2        based on a level of sharing of the physical memory location and whether the IIP associated with  
3        the errant process is precise instruction comprising instructions to:  
4              determine whether the physical memory location is one of a global, a shared and a private  
5        physical memory location;  
6              determine whether the IIP associated with the errant process is precise;  
7              terminate the errant process specified by the precise IIP, if the physical memory location  
8        is private and the IIP is precise; and  
9              determine whether the physical address can be used to indicate the errant process, if the  
10       physical memory location is private and the IIP is imprecise;  
11              terminate the errant process, if the physical address can be used to indicate the errant  
12       process; otherwise, reset the processor.

- 1        16. The machine-readable medium of claim 11, said store the physical address of the  
2       errant process that caused the error instruction comprising an instruction to:  
3              store the physical address of the errant process in a memory register.  
1        17. The machine-readable medium of claim 11, said determining the first virtual  
2       address from the operating system mapping table instruction comprising instructions to:

3           find a physical address entry in the operating system mapping table that matches the  
4       physical address; and

5           read the first virtual address from the matching physical address entry, if the matching  
6       physical address entry is found.

1           18.     The machine-readable medium of claim 11, said determine the second virtual  
2       address from the translation look-aside buffer instruction comprising instructions to:

3           find a physical address entry in the translation look-aside buffer that matches the physical  
4       address; and

5           read the second virtual address from the matching physical address entry, if the matching  
6       physical address entry is found.

1           19.     The machine-readable medium of claim 11, said store the execution instruction  
2       pointer in the interruption instruction pointer instruction comprising an instruction to:

3           store an instruction pointer that existed at the time the error is detected in the interruption  
4       instruction pointer.

1           20. A computer system, comprising:

2           a processor;

3           a system memory coupled to the processor; and

4           a machine-readable medium coupled to the processor, said machine-readable medium

5           having stored thereon a plurality of executable instructions, the plurality of instructions

6           comprising instructions to:

7           detect an error;

8           store a physical address of an errant process that caused the error;

9           store an execution instruction pointer (IP) in an interruption instruction pointer (IIP);

10          determine a first virtual address from an operating system mapping table;

11          determine a second virtual address from a translation look-aside buffer;

12          identify the errant process, if the physical address and the second virtual address match

13          the physical address and the first virtual address.

1           21. The computer system of claim 20 further comprising:

2           an errant process physical address register, said errant process physical address register

3           being configured to receive and store the physical address of the errant process that caused the

4           error.

1           22. The computer system of claim 20 further comprising:  
2           a mapping table, said mapping table being configured to maintain a mapping between all  
3           virtual addresses and all physical addresses in the computer system so that each virtual address is  
4           mapped to one physical address.

1           23. The computer system of claim 22, wherein the mapping table is implemented as  
2           one of:  
3           a buffer; and  
4           a cache array.

5           24. The computer system of claim 20 further comprising:  
6           a translation look-aside buffer (TLB), said TLB being configured to store physical  
7           addresses.

1           25. The computer system of claim 24, the TLB further being configured to store  
2           second virtual addresses associated with the physical addresses.

- 1        26.     The computer system of claim 24 further comprising:
  - 2              a firmware component, said firmware component being configured to provide the
  - 3              physical addresses stored in the TLB.
- 1        27.     The computer system of claim 25, said firmware component further configured to
  - 2              provide the second virtual addresses stored in the TLB.
- 1        28.     The computer system of claim 20, said machine-readable medium comprising:
  - 2              a non-volatile memory.